|  |
| --- |
| *CometPark* |
| Requirements Definition  Document |
| **SE 6387 Advanced Software Engineering Project**  **R.Z. Wenkstern**    ***02/18/2014*** |

|  |
| --- |
| **Group *B 002*** |
| Arunkumar Manickam |
| Hariprasad Natarajan |
| Prasanna Venkatesh |
| Rekha Muthulakshmi |

# 

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Decription** | **Authors** |
| 1.0 | 02/04/2014 | Completed initial draft |  |
| 1.1 | 02/17/2014 | Modified based on the feedback |  |

# Table of Contents

[Revision History 2](#_Toc380447693)

[Table of Contents 3](#_Toc380447694)

[List of figures 5](#_Toc380447695)

[List of tables 6](#_Toc380447696)

[1. Introduction 1](#_Toc380447697)

[2. Overview of the CometPark System: 1](#_Toc380447698)

[3. Features of the CometPark System: 1](#_Toc380447699)

[4. Use Case Diagram: 2](#_Toc380447700)

[5. Components of the System: 2](#_Toc380447701)

[6. Black Box Sequence Diagram 3](#_Toc380447702)

[7. Architecture of the System 3](#_Toc380447703)

[8. Description of the Components: 4](#_Toc380447704)

[Sensor End: 4](#_Toc380447705)

[1. Passive Infrared Sensor: 4](#_Toc380447706)

[2. XBee Series 1 Module: 4](#_Toc380447707)

[3. Battery/Solar Panel: 4](#_Toc380447708)

[Receiver End: 4](#_Toc380447709)

[4. Xbee receiver and Raspberry Pi: 4](#_Toc380447710)

[5. Wi-fi Dongle: 5](#_Toc380447711)

[6. Cloud Server- Amazon Web Services: 5](#_Toc380447712)

[7. Resistor: 5](#_Toc380447713)

[8. Bread board: 5](#_Toc380447714)

[9. Smart Phone: 6](#_Toc380447715)

[Appendix A: Glossary 7](#_Toc380447716)

[Appendix B: References 8](#_Toc380447717)

# List of figures

# List of tables

# 1. Introduction

The purpose of this document is to clearly define the requirements for implementing the CometPark System. The document discusses the various features of the system, the high level architecture, the components needed, the rationale for choosing the components, a table listing the components along with the manufacturer details, price, etc.

# 2. Overview of the CometPark System:

The goal of the CometPark system is to help the users in locating vacant parking spaces on-campus. The application uses sensors to detect the vacant spots and the information is passed via a controller to the server that updates the mobile application periodically.

# 3. Features of the CometPark System:

The following are some of the features of the system:

* Finds nearest vacant parking spot.

The CometPark app locates the nearest available parking spot and guides the user to the spot.

* Remembers where you parked.

Once the user chooses a parking spot, the app marks the spot and can later be used to guide the user to the exact location.

* Works for open and closed parking structures.

The CometPark system has been designed keeping open parking lots in mind, although it should work equally well with closed parking structures.

* Option to show the parking spots associated with the preferred color code.

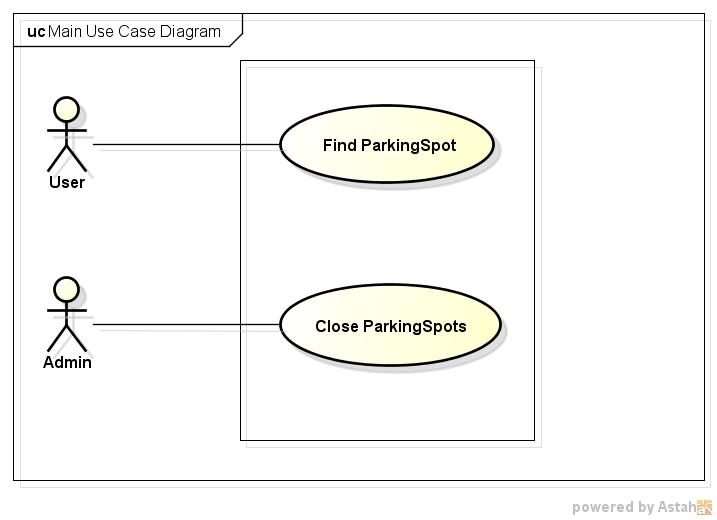
If the user has a parking permit of a particular color, the app can be set to display only the spots associated with that color.

* Notify the closure of all/partial parking lots in case of emergency.

The administrator of the CometPark system has options to send notifications to users in case certain parking lots need to be closed for some occasion or emergency.

# 4. Use Case Diagram:

The following is a high level use-case diagram of the CometPark system.



# 5. Components of the System:

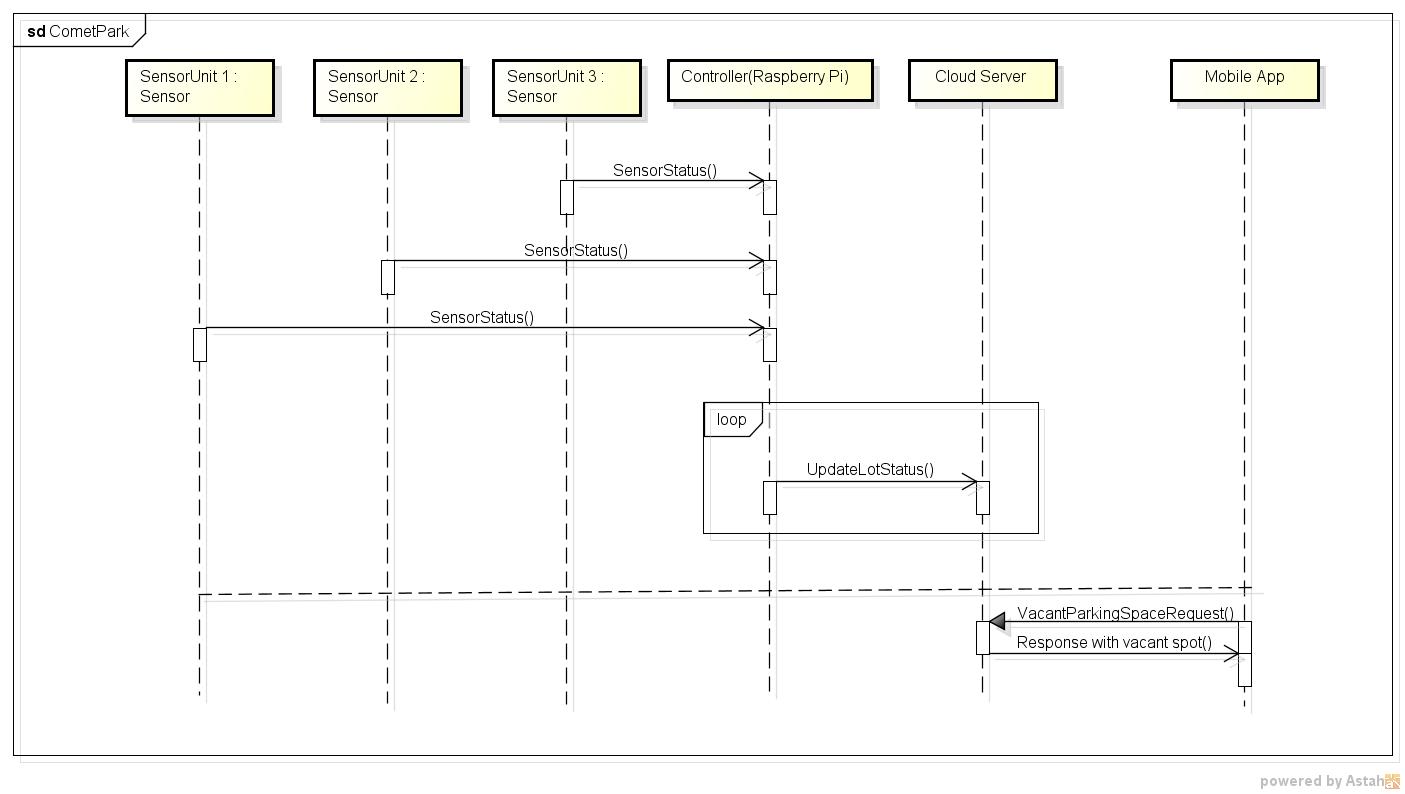
The major components needed to implement the system are as follows:

* Sensor - Passive Infrared Sensor
* Controller – Raspberry Pi
* Cloud - Amazon Web Services
* Smart phone

The following are some of the additional components needed to establish communication between the components and to supply power to the components.

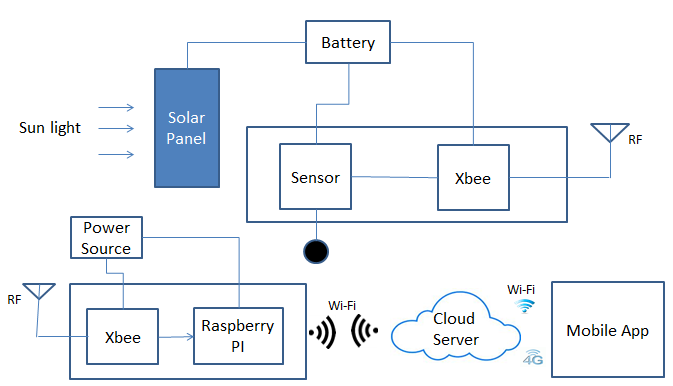
* Xbee Series 1 Module
* Battery
* Wifi dongle
* Resistor
* Bread board

# 6. Black Box Sequence Diagram



# 7. Architecture of the System

The following diagram shows the architecture of the system:



# 8. Description of the Components:

## Sensor End:

The sensor end has the sensor, battery and XBee Series 1 module.

### Passive Infrared Sensor:

Among the various sensors available, we have chosen to use the Passive Infrared sensor since it can be used even in severe weather conditions like snow and due to its affordable price. When a vehicle arrives at the space, the infrared sensor detects the presence of the vehicle and sends the signal through the Xbee transmitter.

|  |  |  |  |
| --- | --- | --- | --- |
| Features | Passive Infrared Sensor | Ultrasonic Sensor | Magnetometer |
| Voltage | 5V to 20V | 5V | 3.6V |
| Power | 65 mA | less than 2mA | 100 μA |
| Range | 120 degree | 15 degrees | Wide range |
| Temperature | – 15 ~ +70 | – 15 ~ +70 | -30 ~ +85 |
| Detection distance | 7 m | 2cm-450cm | -8 ~ +8 gauss |
| Compatible with open space? | Yes | No | Yes |
| Cost | Cheap | Cheap | Costly |

Specification comparison with other sensors

As our project must be compatible for both open and closed parking spaces, PIR and Magnetometer are preferred. But considering cost, PIR was selected for our project.

### XBee Series 1 Module:

The communication between the sensor and the Controller is achieved through the Xbee module. The Xbee Series 1 is a wireless communication device that uses the Zigbee protocol which is based on the IEEE 802.15.4-2003 standard for wireless personal area networks (WPANs). It is mainly used for applications with RF transmission and operates with 3.3V and consumes 50Ma. The reason for choosing the Xbee module for transmission is that it each module can be set to a unique ID and also does not require a microcontroller to transmit the data from the sensor to the Raspberry Pi.

### Battery/Solar Panel:

The sensor and the Xbee are powered by a battery pack. This setup is for the purpose of demo. When implementing for a number of parking spaces, a Solar Panel along with a rechargeable battery pack as a backup can be used for power supply.

## Receiver End:

### Xbee receiver and Raspberry Pi:

The receiver end has another Xbee receiver mo1dule connected to a Raspberry Pi controller. This Xbee receiver receives the signal sent from various Xbee transmitters which is processed by the Raspberry Pi controller. Raspberry Pi has different models and the model chosen for this project is ‘Raspberry Pi- Model B’ since its specifications are best suited for the system’s requirements.

#### Specifications of Raspberry Pi Model B:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Chip** | |  |  | | --- | --- | | |  | | --- | | Broadcom BCM2835 SoC full HD multimedia applications processor | | |  | |
| **CPU** | 700 MHz Low Power ARM1176JZ-F Applications Processor |
| **GPU** | Dual Core VideoCore IV® Multimedia Co-Processor |
| **Memory** | 512MB SDRAM |
| **Ethernet** | onboard 10/100 Ethernet RJ45 jack |
| **USB 2.0** | Dual USB Connector |
| **Onboard Storage** | SD, MMC, SDIO card slot |
| **Operating System** | Linux |

### Wi-fi Dongle:

The sensors status from the Raspberry Pi controller is sent to the Cloud Server through a Wi-Fi network. A Wi-Fi dongle is attached to the Raspberry Pi to enable it to connect to a wireless network.

### Cloud Server- Amazon Web Services:

For the purpose of this project, we have planned to host the application on the Amazon Cloud. The Amazon web service offers 750 hours of free usage for each account and is billed after that depending on the usage. The detailed pricing information of the cloud service can be found in the following link:

<http://aws.amazon.com/ec2/pricing/>

For future implementation of the project on a large scale, hosting the application on the UTD Server is another option.

### Resistor:

In order to regulate the voltage variations between the various devices like the sensor, battery, resistors are used.

### Bread board:

A bread board is required as the base board for making the circuit connections on sensor side.

### Smart Phone:

A mobile phone that has the app installed is used to get the most updated information to locate the available parking space.

**COMPONENTS WITH THEIR MANUFACTURERS AND PRICE:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Model** | **Manufacturer** | **Price** |
| Passive Infrared Sensor | HC-SR501 | Amazon | $8.45 for 5 sensors |
| Xbee Module | Series 1 | RobotMesh | $20.23 |
| Controller | Model B | Raspberry Pi | $35.00 |
| Cloud Server | AWS | Amazon | Free for 750 hrs |
| Wifi Adapter | EW-7811Un | Edimax | $9.99 |
| Battery Pack | 6V – DiaMec | Apex Battery | $4.75 |
| Bread board | 400 tie-points | Amazon | $1.9 |

Based on our proposed components, the following is an estimate of the cost per parking space:

Cost per parking space: $28.57

Cost per controller component: $69.97

# Appendix A: Glossary

|  |  |
| --- | --- |
| **Term** | **Definition** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# Appendix B: References